arXiv:cond-mat/9709062v1 [cond-mat.soft] 4 Sep 1997

IMPROVING THE HOPFIELD-TANK APPROACH
FOR THE TRAVELING SALESMAN PROBLEM

M. Argollo de Menezes and T.J.P.Penna

Instituto de Fisica, Universidade Federal Fluminense

e-mail: marcio@if.uff.br and tjpp@if.uff.br

Av. Litorânea s/n,24210-340, Niterói, RJ, Brasil

In this work we revisit the Hopfield-Tank algorithm for the traveling salesman problem (TSP) [1] and report encouraging results, with a different dynamics, that makes the algorithm more efficient finding better solutions in much less computational time.

Keywords: neural networks, optimization problems, traveling salesman problem, Kawasaki dynamics

The traveling salesman problem (TSP) is the problem of finding the shortest path needed for a salesman to visit N cities on a planar surface. It belongs to a class of problems called non-deterministic polynomial (NP) complete that have many applications to physics, mainly because of the methods developed to solve them. The computational complexity of the TSP problem resides in the rapidly increasing number of possible configurations with the number of cities. For example, for an N-cities problem, the number of possible configurations is $(N-1)!/2$, which is of order of 10^{1217} for $N = 532$, therefore it is of vital importance to develop some heuristics to solve the problem and find near optimal solutions.

Among the methods employed to solve the TSP problem, there exists neural-networks (NN) [1][3], simulated annealing (SA) [2], genetic algorithms (GA) [4], Lin-Kernighan elastic algorithm [5], real-space renormalization [9] and others.

First proposed by Hopfield and Tank [1] in 1985, the neural-network realization of the TSP problem is of particular interest because of its "silicon implementation". However in other article, Wilson and Pawley [6] discussed the stability of the method and reported unsatisfactory results even with some modifications on the original algorithm. In this paper we revisited this problem, and show that a Kawasaki dynamics of updating spin variables of the network is highly efficient in this problem. This article is divided in the following sections: in section two we summarize the Hopfield-Tank model. In section three the computational implementation of both the original algorithm and the new one are discussed and results are presented in section four.

1. Hopfield neural-networks

In a neural network we have the neurons that have a certain degree of activation, depending on how and with which intensity they are interconnected [1]. This connection between the neurons are called synapses, and they are not physical, but chemical. The degree of activation of a particular pair of neurons is proportional to the amount of neurotransmitter released by the neuron. This neurotransmitter allows the electric pulse generated by some stimulus to pass from one neuron to others, transmitting the information neuron by neuron until it reaches the brain. The neuron dynamics will be governed by the state of other neurons which it is connected to by the synapses.

Among the many neural network models developed to simulate some features of human brain like recognition and learning, Hopfield [7] proposed a neuron dynamics which minimize the following energy function

$$H = -\sum d_{ij}\sigma_i\sigma_j.$$

where σ_i is the state of the ith neuron of the network and d_{ij} is the synaptic strength between neurons i and j. The dynamics which minimizes this energy function is

$$\sigma_i(t+1) = \text{sgn}(\sum_j d_{ij}\sigma_j)$$

One first simplification from the behaviour of real neurons is to set the state of the neuron as a Boolean variable, that means, it is activated or not ($\sigma_i = -1, 1$). One particular feature of Hopfield-like neural networks is that they can modify their behaviour in response to their environment (feedback mechanism),that means, given a set of inputs to the neurons, they self-adjust to produce consistent responses. In order to simulate the features of the human brain, one has to implement a learning stage. One of the most used training algorithms is the Hebb rule, which is an unsupervised learning where the synaptic strength between two neurons is increased when both the source and destination neuron are activated. We can write this as $d_{ij} = \sum_\mu^N \xi_i^\mu \xi_j^\mu$,where ξ^μ is one of the N patterns stored in the brain. We represent one pattern by a set of N neurons in a particular configuration.

This learning stage is appropriate for pattern recognition by a neural network, which means searching for a local minimum in the energy surface of H, which is different of the problem of optimization, where the search is for the global minimum.

As this network device works as a multiple-feedback circuit, which is not governed by any variable parameter, like the artificial temperature in SA or mutations in GA, it is claimed as "silicon implementable", that is, it is possible to develop microchips that execute this function.

The realization of the traveling salesman problem in Hopfield neural-networks consists on attributing to the distance between two cities A and B a symmetric synaptic strength d_{ab} and so on to all pairs of cities on a unit square (we normalize the maximum distance, and then the solutions given can be compared with any others with no need to rescaling). The problem is mapped onto a network by representing each city by a row of N elements, where N is the number of cities.

Then, the output of a neuron, σ_{ij} (called spin variable), means the *ith* city visited in the *jth* order. If it is set to one, the city is really visited in that order, if it is set to zero, then this assertion is false.
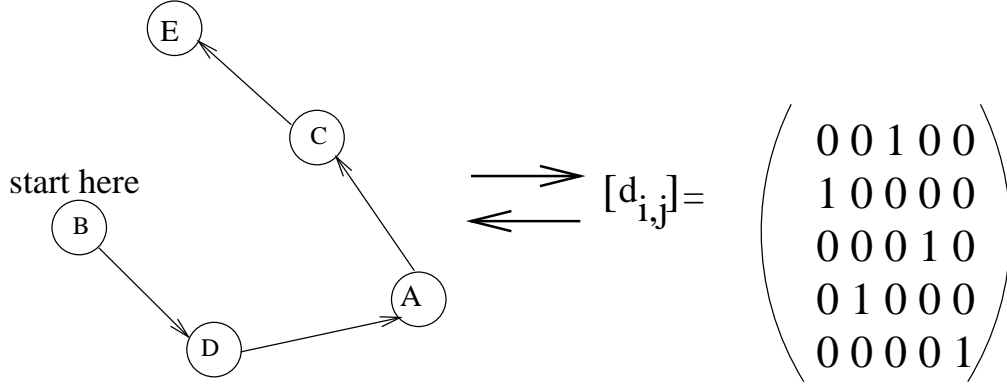

Fig. 1. Matrix representation of the TSP problem

Fig. 1 is an example where the city B is visited first, followed by D,A,C and E. The resulting perimeter, or energy, is $d_{B,D} + d_{D,A} + d_{C,E} + d_{E,B}$. It can be verified that this $N \times N$ matrix is sparse (more exactly, it has only N elements).

The way of defining the energy function is not unique and requires a specification of the problem and what is to be minimized. In the TSP case, it is required that each row and each column have only one non-zero element (which means that a city is to be visited only once and only one city per time is visited), and the basic requirement is that solutions with short paths must be favoured. One satisfactory energy function proposed in [1] is

$$E = \frac{A}{2} \sum_{X}^{N} \sum_{i}^{N} \sum_{j\neq i}^{N} \sigma_{Xi}\sigma_{Xj}$$

$$+\frac{B}{2} \sum_{i}^{N} \sum_{X}^{N} \sum_{Y\neq X}^{N} \sigma_{Xi}\sigma_{Yi}$$

$$+\frac{C}{2} [(\sum_{X}^{N} \sum_{i}^{N} \sigma_{Xi}) - N]^2$$

$$+\frac{D}{2} \sum_{X}^{N} \sum_{Y\neq X}^{N} \sum_{i}^{N} d_{XY}\sigma_{Xi}(\sigma_{yi+1} + \sigma_{yi-1})$$

with A, B, C and $D > 0$. The first and second terms vanish only when there is one and only one non-zero element in each row and column, respectively. The third term is zero when there are just N of these non-null elements in the matrix, avoiding the convergence to the situation where all elements are zero, and the last term represent

the length of any valid tour, and favours the shortest tours. The output of a neuron can be considered as a continuous variable, as in the original model, or discrete and equal to 0 or 1, as in this study, which is more efficient to deal with because of the well known computer tricks associated with these Boolean variables [10].

In their 1985 paper [1], Hopfield and Tank reported success for a 30 cities problem, a fact which is contested by Wilson and Pawley three years later [1], who say that the convergence of the network to an acceptable tour, which means a tour with all constraints satisfied, is difficult to reach and that this possibility decreases rapidly with the number of cities (in fact, for a 10 cities problem, after 1000 iterations of the network an acceptable tour was obtained in only 8% of

all cases). They also tried to improve the algorithm in three ways: by giving excessively large values to impossible spin configurations like σ_{ii}, varying the values of the energy parameters, and changing the initial conditions. Even with all this changes the final result was not better than the original one, and since then interest on Hopfield NN algorithm for TSP has decreased. With the new trick shown in this paper we shed a new light on this problem.

2. Computational Implementation

In this section we will discuss the computational implementation of the problem, the properties of recurrent networks, like stability, and the improvement of the Hopfield-Tank algorithm for the traveling salesman problem by introducing the Kawasaki dynamics.

As cited on the previous section, Hopfield neural networks are said to be recurrent , because of the dynamical feedback mechanism of changing the input as a function of the output of the network. For a stable network, after successive iterations the changes on the output will become smaller and smaller until it reaches an equilibrium state where all the outputs will become constant in the successive iterations. A network can be proven to be stable [8] if the synaptic matrix is symmetric with zero

s on its main diagonal, that means, $d_{ij} = d_{ji}$, and $d_{ii} = 0$. But because of the highly irregular form of the TSP energy surface, there is no guarantee that it will converge to the global rather then to a local minimum, and even whether the minimum found will respect all the constraints (providing an acceptable tour) or not. The way to deal with it is to make a statistics of the N cities problem.

With the constraints imposed, we can randomly change the state of a neuron and calculate the new energy, accepting the new configuration if it leads to a lower energy. In order to find minimal solutions we apply a parallel dynamics on the algorithm, that means, we change the state of a neuron and calculate the new energy, if it decreases we accept the change, if not, we invert it again. But the trick is to test all neurons at the same iteration, changing from the old configuration to the state obtained

after all neuron have been tested and updated, and the results found are better then when updating sequentially: one single neuron at a time. We consider one

time step when the configuration of the network doesn't change after $10 \times N$ iterations, where N is the number of cities, or if it does not converge after $100 \times N$ iterations. This technique is slow, given the number of possible configurations of the spins, namely 2^{N^2}, which increases enormously with the number of cities. Also, the number of operations required to determine the energy of a configuration is rather high.

What we can do to simplify this problem is try to reduce the number of operations of the algorithm. Our proposal is to update spin variables according to the Kawasaki dynamics. The Kawasaki dynamics consists in choosing two neurons and checking if their exchange will decrease the energy value. In the particular case of the TSP it can violate the constraints. Therefore, if we select only active neurons, we can start a simulation from a situation where there is only one active neuron in each column and row (like, for instance, the identity matrix), which means a valid tour. With this, we are able to always interchange two rows at a time, thus always having a valid tour as output and overcoming the problem of convergence. This task reduces the number of operations required to calculate the energy function, since we do not need the constraints anymore, and the computational time required to obtain a desired number of valid tours is greatly reduced.

3. Numerical Results and Conclusions

In order to have reliable results we make a statistics of the possible energy configurations (perimeter of the valid tours) of one city set, or make a statistics of the number of convergences to a valid tour for various randomly chosen city sets. As the number of acceptable tours increases with the number of cities, make a statistics of the possible energy configurations for a single city set is appropriate only when is N large . A statistics of the number of convergences for a valid tour can give good results in comparing the efficiency of different dynamics not for one single particular city set, but for as

many sets as one can build. In this paper we will use the second statistics for 5000 different sets.

Simulating 5000 different city sets we can construct histograms for the final configurations obtained with both the traditional and Kawasaki dynamics and observe that this new task is able to find more near-optimal solutions and paths even shorter than the shortest path found in the traditional way, which explains the larger dispersion for the Kawasaki dynamics. It can be seen that it obeys a Gaussian distribution in both cases and that the number of possible states is greater for $N = 20$.

The other important feature of this method, the computational efficiency, is shown in fig.3, where the time is in seconds and was obtained in a Pentium 133 MHz, with the program in C code compiled with GNU C compiler, for 200 sets . The results are better for N large, which is the region of real interest. The fact that the difference between the histograms of the two dynamics is not so significant for $N = 20$ is not disappointing, since the shortest path is always achieved with the
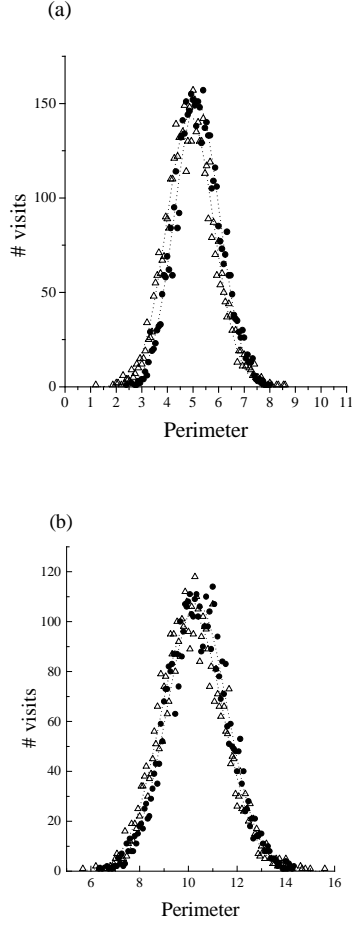
(a)

(b)

Fig. 2. Histograms of number of convergences vs. perimeter (P) of the tour (energy) for **(a)** 10 cities and **(b)** 20 cities. Bullets are for Hopfield-Tank dynamics and triangles are for Kawasaki dynamics. Gaussian fits for the histograms give $\bar{P}_{HT} = 5.15$ and $\bar{\Delta} P^2_{HT} = 1.77$ and $\bar{P}_K = 4.88$ and $\bar{\Delta} P^2_K = 1.85$ for $N = 10$. For $N = 20$ we have $\bar{P}_{HT} = 10.36$ and $\bar{\Delta} P^2_{HT} = 2.48$ and $\bar{P}_K = 10.15$ and $\bar{\Delta} P^2_K = 2.58$.

Kawasaki dynamics and the time required for constructing the histograms are very different.
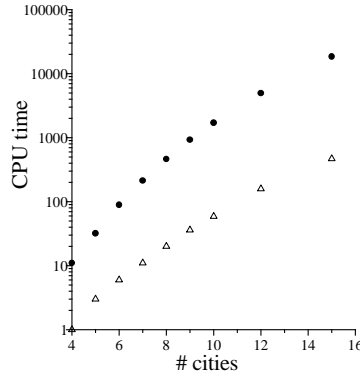

Fig. 3. Computation time vs. number of cities for Hopfield-Tank (•) and Kawasaki dynamics (△). Note the logarithm scale in the vertical axis.

From the results shown above, it can be seen that this dynamics, using multi-spin coding techniques [10], makes it worth applying the Hopfield NN algorithm to systems with sizes of real interest.

4. Acknowledgements

1. Hopfield,J.J.,Tank,D.W., Biol. Cybern. **52**, 141-152 (1985).
2. Kirkpatrick,S.,Gelatt,C.D.,Vecchi,M.P., Science, **220**, 671-680 (1983).
3. T.J.P.Penna, Phys. Rev. **E 51**, R1 (1995).
4. Brady, R.M., Nature **317**, 804-806 (1985).
5. Lin, S. and Kernighan, B.W., Oper. Res., **21**,498 (1973).
6. Wilson, G.V. and Pawley, G.S., Biol. Cybern., **58**, 63-70 (1988).
7. J.J. Hopfield, Proc.Nat.Ac.Sci., **79**, 2554-58 (1982).
8. P.D. Wasserman, *Neural Computing - theory and practice*, Van Nostrand Reinhold, New York,(1989).
9. Yoshiyuki,U. and Yoshiki,K., Phys. Rev. Lett., **75**, 1684 (1995).
10. de Oliveira,P.M.C., *Computing Boolean Statistical Models*, World Scientific, Singapore, ISBN 981-02-0238-5 (1991).